Programming Assignment

Said HASHEMI

Generics were introduced to Java to provide stronger compile-time type checking and eliminate many runtime ClassCastExceptions by allowing classes and methods to be parameterized by type (Divertitto, 2022). Parameterized types improve flexibility and reusability: the same GenericCatalog implementation can manage Book, DVD, Magazine or any future LibraryItem subtype without code duplication (Eck, 2022; Kumar, 2023). Within the Java Collections Framework, generics enable type-safe collections (e.g., List<Book>), improving program correctness and readability (Eck, 2022). Tutorials and videos (e.g., Bro Code) reinforce these practical patterns for generic classes and methods (Bro Code, 2020).

Code

```
/**
 * Main.java
 *
 * Generic Library Catalog assignment (CS 1103-01 - AY2026-T1)
 * - Demonstrates:
 *   * Generic LibraryItem<ID>
 *   * GenericCatalog<T extends LibraryItem<ID>, ID>
 *   * CLI menu + automated test scenarios
 */

import java.util.*;

/**
 * Generic LibraryItem class.
 * @param <ID> type of the itemID (Integer, String, UUID, etc.)
 */
class LibraryItem<ID> {
    private ID itemID;
```

```
    private String title;
    private String author; // For DVDs this could be "director"

    public LibraryItem(ID itemID, String title, String author) {
        this.itemID = itemID;
        this.title = title;
        this.author = author;
    }

    public ID getItemID() { return itemID; }
    public String getTitle() { return title; }
    public String getAuthor() { return author; }

    public void setItemID(ID itemID) { this.itemID = itemID; }
    public void setTitle(String title) { this.title = title; }
    public void setAuthor(String author) { this.author = author; }

    @Override
    public String toString() {
        return String.format("ID: %s | Title: %s | Author/Creator: %s",
                String.valueOf(itemID), title, author);
    }
}

/**
 * GenericCatalog class storing items keyed by their ID.
 * T must extend LibraryItem<ID>.
 *
 * @param <T>  item type
 * @param <ID> item ID type
 */
class GenericCatalog<T extends LibraryItem<ID>, ID> {
```

```
    private final Map<ID, T> items;


    public GenericCatalog() {
        this.items = new HashMap<>();
    }


    /**
        * Adds an item to the catalog. If an item with same ID exists, it will be replaced.
     * @param item item to add
     * @return true if new item added, false if it replaced an existing item
     */
    public boolean addItem(T item) {
        boolean isNew = !items.containsKey(item.getItemID());
        items.put(item.getItemID(), item);
        return isNew;
    }


    /**
     * Removes an item by ID.
     * Throws NoSuchElementException if ID not present.
     * @param id item ID to remove
     * @throws NoSuchElementException when id not present
     */
    public void removeItem(ID id) {
        if (!items.containsKey(id)) {
            throw new NoSuchElementException("Item with ID '" + id + "' does not exist in the catalog.");
        }
        items.remove(id);
    }


    /**
```

```
     * Retrieves item by ID.
     * @param id item ID
     * @return item or null if not found
     */
    public T getItem(ID id) {
        return items.get(id);
    }

    /**
     * Returns a collection of all items in the catalog.
     * @return Collection<T>
     */
    public Collection<T> listItems() {
        return Collections.unmodifiableCollection(items.values());
    }

    /**
     * Returns number of items stored.
     */
    public int size() {
        return items.size();
    }
}

/* --------- Example concrete subclasses to demonstrate variety --------- */

/** Book with Integer ID */
class Book extends LibraryItem<Integer> {
    private String isbn; // optional field to enrich the example

    public Book(Integer id, String title, String author, String isbn) {
        super(id, title, author);
```

```
        this.isbn = isbn;
    }

    public String getIsbn() { return isbn; }
    public void setIsbn(String isbn) { this.isbn = isbn; }

    @Override
    public String toString() {
        return String.format("Book | %s | ISBN: %s", super.toString(), isbn);
    }
}

/** DVD with String ID (maybe a code) */
class DVD extends LibraryItem<String> {
    private int durationMinutes;

    public DVD(String id, String title, String director, int durationMinutes) {
        super(id, title, director);
        this.durationMinutes = durationMinutes;
    }

    public int getDurationMinutes() { return durationMinutes; }
    public void setDurationMinutes(int d) { this.durationMinutes = d; }

    @Override
    public String toString() {
            return String.format("DVD  | %s | Duration: %d mins", super.toString(),
durationMinutes);
    }
}

/* ---------------------- Main & CLI + Tests ---------------------- */
```

```
public class Main {

    private static final Scanner scanner = new Scanner(System.in);

    public static void main(String[] args) {
        // Create two catalogs to demonstrate generics handling different ID types / item types
        GenericCatalog<Book, Integer> bookCatalog = new GenericCatalog<>();
        GenericCatalog<DVD, String> dvdCatalog = new GenericCatalog<>();

        // Pre-populate with some items for demonstration
            bookCatalog.addItem(new Book(1001, "Clean Code", "Robert C. Martin", "978-0132350884"));
        bookCatalog.addItem(new Book(1002, "Effective Java", "Joshua Bloch", "978-0134685991"));

            dvdCatalog.addItem(new DVD("D-001", "The Matrix", "The Wachowskis", 136));
            dvdCatalog.addItem(new DVD("D-002", "Inception", "Christopher Nolan", 148));

        // Run automated tests (demonstration/testing requirement)
        System.out.println("=== Running automated test scenarios ===");
        runTests(bookCatalog, dvdCatalog);

        // Then enter interactive CLI for user
        System.out.println("\n=== Entering interactive catalog CLI ===");
        runCLI(bookCatalog, dvdCatalog);
    }

    /**
     * Demonstrates test scenarios:
     * - add items
```

```
    * - remove existing
    * - attempt to remove non-existent (error handling)
    * - list contents
    */
    private static void runTests(GenericCatalog<Book, Integer> bookCatalog, GenericCatalog<DVD, String> dvdCatalog) {
        System.out.println("\n-- Book catalog initial contents --");
        printCatalog(bookCatalog);

        System.out.println("\nAdding a new book (ID 1003)...");
        Book b = new Book(1003, "Java Generics in Practice", "Some Author", "ISBN-9999");
        boolean added = bookCatalog.addItem(b);
        System.out.println("Added new? " + added);
        printCatalog(bookCatalog);

        System.out.println("\nRemoving book with ID 1002 (should succeed)...");
        try {
            bookCatalog.removeItem(1002);
            System.out.println("Removed 1002 successfully.");
        } catch (NoSuchElementException e) {
            System.out.println("Error: " + e.getMessage());
        }
        printCatalog(bookCatalog);

        System.out.println("\nAttempting to remove book with ID 5555 (non-existent) to test error handling...");
        try {
            bookCatalog.removeItem(5555);
            System.out.println("Unexpected: removal succeeded.");
        } catch (NoSuchElementException e) {
            System.out.println("Caught expected exception: " + e.getMessage());
        }
```

```
        System.out.println("\n-- DVD catalog initial contents --");
        printCatalog(dvdCatalog);

        System.out.println("\nAdding a new DVD D-003 and listing...");
          dvdCatalog.addItem(new DVD("D-003", "Interstellar", "Christopher Nolan",
169));
        printCatalog(dvdCatalog);

        System.out.println("\nTrying to get DVD D-002 details:");
        DVD found = dvdCatalog.getItem("D-002");
        System.out.println(found != null ? found.toString() : "D-002 not found");

        System.out.println("\nAutomated tests finished.");
    }

    /**
     * Simple interactive CLI allowing user to manipulate both catalogs.
     */
        private static void runCLI(GenericCatalog<Book, Integer> bookCatalog,
GenericCatalog<DVD, String> dvdCatalog) {
        boolean quit = false;
        while (!quit) {
            System.out.println("\nChoose catalog to work with:");
            System.out.println("1. Book catalog (Integer IDs)");
            System.out.println("2. DVD catalog (String IDs)");
            System.out.println("3. Exit");
            System.out.print("Your choice: ");
            String choice = scanner.nextLine().trim();

            switch (choice) {
                case "1":
                    handleBookCatalog(bookCatalog);
```

```
                break;
            case "2":
                handleDVDcatalog(dvdCatalog);
                break;
            case "3":
                quit = true;
                break;
            default:
                System.out.println("Invalid choice. Try 1, 2 or 3.");
        }
    }
    System.out.println("Exiting CLI. Goodbye!");
  }


    private static void handleBookCatalog(GenericCatalog<Book, Integer>
catalog) {
    System.out.println("\n-- Book Catalog Menu --");
    System.out.println("a: List all books");
    System.out.println("b: Add a book");
    System.out.println("c: Remove a book by ID");
    System.out.println("d: Get book details by ID");
    System.out.println("e: Back");
    System.out.print("Choice: ");
    String c = scanner.nextLine().trim().toLowerCase();
    switch (c) {
        case "a":
            printCatalog(catalog);
            break;
        case "b":
            try {
                System.out.print("Enter integer ID: ");
                Integer id = Integer.valueOf(scanner.nextLine().trim());
```

```
                System.out.print("Enter title: ");
                String title = scanner.nextLine().trim();
                System.out.print("Enter author: ");
                String author = scanner.nextLine().trim();
                System.out.print("Enter ISBN (optional): ");
                String isbn = scanner.nextLine().trim();
                Book book = new Book(id, title, author, isbn);
                boolean added = catalog.addItem(book);
                  System.out.println(added ? "Book added." : "Book replaced existing item with same ID.");
            } catch (NumberFormatException nf) {
                System.out.println("Invalid ID. Must be an integer.");
            }
            break;
        case "c":
            try {
                System.out.print("Enter integer ID to remove: ");
                Integer remId = Integer.valueOf(scanner.nextLine().trim());
                catalog.removeItem(remId);
                System.out.println("Removed item with ID " + remId);
            } catch (NumberFormatException nfe) {
                System.out.println("Invalid ID format.");
            } catch (NoSuchElementException e) {
                System.out.println("Error: " + e.getMessage());
            }
            break;
        case "d":
            try {
                System.out.print("Enter integer ID to get: ");
                Integer g = Integer.valueOf(scanner.nextLine().trim());
                Book found = catalog.getItem(g);
                  System.out.println(found != null ? found.toString() : "No book found with ID " + g);
```

```
            } catch (NumberFormatException nfe) {
                System.out.println("Invalid ID.");
            }
            break;
        case "e":
            // back
            break;
        default:
            System.out.println("Invalid choice.");
    }
}

private static void handleDVDcatalog(GenericCatalog<DVD, String> catalog) {
    System.out.println("\n-- DVD Catalog Menu --");
    System.out.println("a: List all DVDs");
    System.out.println("b: Add a DVD");
    System.out.println("c: Remove a DVD by ID");
    System.out.println("d: Get DVD details by ID");
    System.out.println("e: Back");
    System.out.print("Choice: ");
    String c = scanner.nextLine().trim().toLowerCase();
    switch (c) {
        case "a":
            printCatalog(catalog);
            break;
        case "b":
            System.out.print("Enter string ID (e.g. D-010): ");
            String id = scanner.nextLine().trim();
            System.out.print("Enter title: ");
            String title = scanner.nextLine().trim();
            System.out.print("Enter director: ");
            String director = scanner.nextLine().trim();
```

```
                System.out.print("Enter duration (minutes): ");
                try {
                    int duration = Integer.parseInt(scanner.nextLine().trim());
                    DVD dvd = new DVD(id, title, director, duration);
                    boolean added = catalog.addItem(dvd);
                        System.out.println(added ? "DVD added." : "DVD replaced existing item with same ID.");
                } catch (NumberFormatException nfe) {
                    System.out.println("Invalid number for duration.");
                }
                break;
            case "c":
                System.out.print("Enter string ID to remove: ");
                String rem = scanner.nextLine().trim();
                try {
                    catalog.removeItem(rem);
                    System.out.println("Removed item with ID " + rem);
                } catch (NoSuchElementException e) {
                    System.out.println("Error: " + e.getMessage());
                }
                break;
            case "d":
                System.out.print("Enter string ID to get: ");
                String getId = scanner.nextLine().trim();
                DVD found = catalog.getItem(getId);
                    System.out.println(found != null ? found.toString() : "No DVD found with ID " + getId);
                break;
            case "e":
                // back
                break;
            default:
                System.out.println("Invalid choice.");
```

```
        }
    }


    // Utility: print catalog (works for any T extends LibraryItem<ID>)
                private static <T extends LibraryItem<ID>, ID> void
printCatalog(GenericCatalog<T, ID> catalog) {
        System.out.println("Catalog size: " + catalog.size());
        if (catalog.size() == 0) {
            System.out.println("(empty)");
            return;
        }
        for (T item : catalog.listItems()) {
            System.out.println(" - " + item.toString());
        }
    }
}
```

Output Screenshots

```
=== Running automated test scenarios ===

-- Book catalog initial contents --
Catalog size: 2
 - Book | ID: 1001 | Title: Clean Code | Author/Creator: Robert C. Martin | ISBN: 978
     -0132350884
 - Book | ID: 1002 | Title: Effective Java | Author/Creator: Joshua Bloch | ISBN: 978
     -0134685991

Adding a new book (ID 1003)...
Added new? true
Catalog size: 3
 - Book | ID: 1001 | Title: Clean Code | Author/Creator: Robert C. Martin | ISBN: 978
     -0132350884
 - Book | ID: 1002 | Title: Effective Java | Author/Creator: Joshua Bloch | ISBN: 978
     -0134685991
 - Book | ID: 1003 | Title: Java Generics in Practice | Author/Creator: Some Author |
     ISBN: ISBN-9999

Removing book with ID 1002 (should succeed)...
Removed 1002 successfully.
Catalog size: 2
 - Book | ID: 1001 | Title: Clean Code | Author/Creator: Robert C. Martin | ISBN: 978
     -0132350884
```

```
Attempting to remove book with ID 5555 (non-existent) to test error handling...
Caught expected exception: Item with ID '5555' does not exist in the catalog.

-- DVD catalog initial contents --
Catalog size: 2
 - DVD  | ID: D-002 | Title: Inception | Author/Creator: Christopher Nolan | Duration
     : 148 mins
 - DVD  | ID: D-001 | Title: The Matrix | Author/Creator: The Wachowskis | Duration:
     136 mins

Adding a new DVD D-003 and listing...
Catalog size: 3
 - DVD  | ID: D-003 | Title: Interstellar | Author/Creator: Christopher Nolan |
     Duration: 169 mins
 - DVD  | ID: D-002 | Title: Inception | Author/Creator: Christopher Nolan | Duration
     : 148 mins
 - DVD  | ID: D-001 | Title: The Matrix | Author/Creator: The Wachowskis | Duration:
     136 mins

Trying to get DVD D-002 details:
DVD  | ID: D-002 | Title: Inception | Author/Creator: Christopher Nolan | Duration:
    148 mins

Automated tests finished.
```

```
=== Entering interactive catalog CLI ===

Choose catalog to work with:
1. Book catalog (Integer IDs)
2. DVD catalog (String IDs)
3. Exit
Your choice: 1

-- Book Catalog Menu --
a: List all books
b: Add a book
c: Remove a book by ID
d: Get book details by ID
e: Back
Choice: b
Enter integer ID: 1003
Enter title: Book for me
Enter author: Said
Enter ISBN (optional): 1243567891011
Book replaced existing item with same ID.
```

```
Choose catalog to work with:
1. Book catalog (Integer IDs)
2. DVD catalog (String IDs)
3. Exit
Your choice: 2

-- DVD Catalog Menu --
a: List all DVDs
b: Add a DVD
c: Remove a DVD by ID
d: Get DVD details by ID
e: Back
Choice: C
Enter string ID to remove: D-001
Removed item with ID D-001
```

```
Choose catalog to work with:
1. Book catalog (Integer IDs)
2. DVD catalog (String IDs)
3. Exit
Your choice: 1

-- Book Catalog Menu --
a: List all books
b: Add a book
c: Remove a book by ID
d: Get book details by ID
e: Back
Choice: d
Enter integer ID to get: 1001
Book | ID: 1001 | Title: Clean Code | Author/Creator: Robert C. Martin | ISBN: 978
    -0132350884
```

Conclusion

Generics reduce code duplication by allowing a single GenericCatalog implementation to handle items with different underlying ID types and different item subclasses. Because types are parameterized at compile-time, many class-cast errors are prevented earlier (Divertitto, 2022; Eck, 2022). Bounded types (T extends LibraryItem<ID>) allow sensible constraints while preserving flexibility (Kumar, 2023). The example program above uses these principles to create a modular, reusable, and type-safe catalog system (Bro Code, 2020).

References

- Divertitto, A. (2022, August 18). *Java generics: how to use angled brackets in practice*. CodeGym. https://codegym.cc/groups/posts/generics-in-java. (Discusses compile-time checking, generic classes/methods, wildcards.) (Divertitto, 2022)
- Eck, D. J. (2022). *Introduction to programming using java version 9, JavaFX edition.* Chapter 10 – Generic Programming and Collection Classes. Licensed under CC 4.0. https://math.hws.edu/javanotes/ (Covers generics, collection classes, writing generic classes and methods.) (Eck, 2022)
- Kumar, A. (2023, April 18). *Mastering generics in Java: A comprehensive guide for Java developers*. Tech Thoughts Explorer. https://techthoughtsexplorer.hashnode.dev/mastering-generics-in-java-a-comprehensive-guide-for-java-developers (Explains bounded type parameters, wildcards, and examples.) (Kumar, 2023)
- Bro code. (2020, July 27). *Java generics* [Video]. YouTube. https://youtu.be/jUcAyZ5OUm0 (Video tutorial showing practical generics usage.) (Bro Code, 2020)